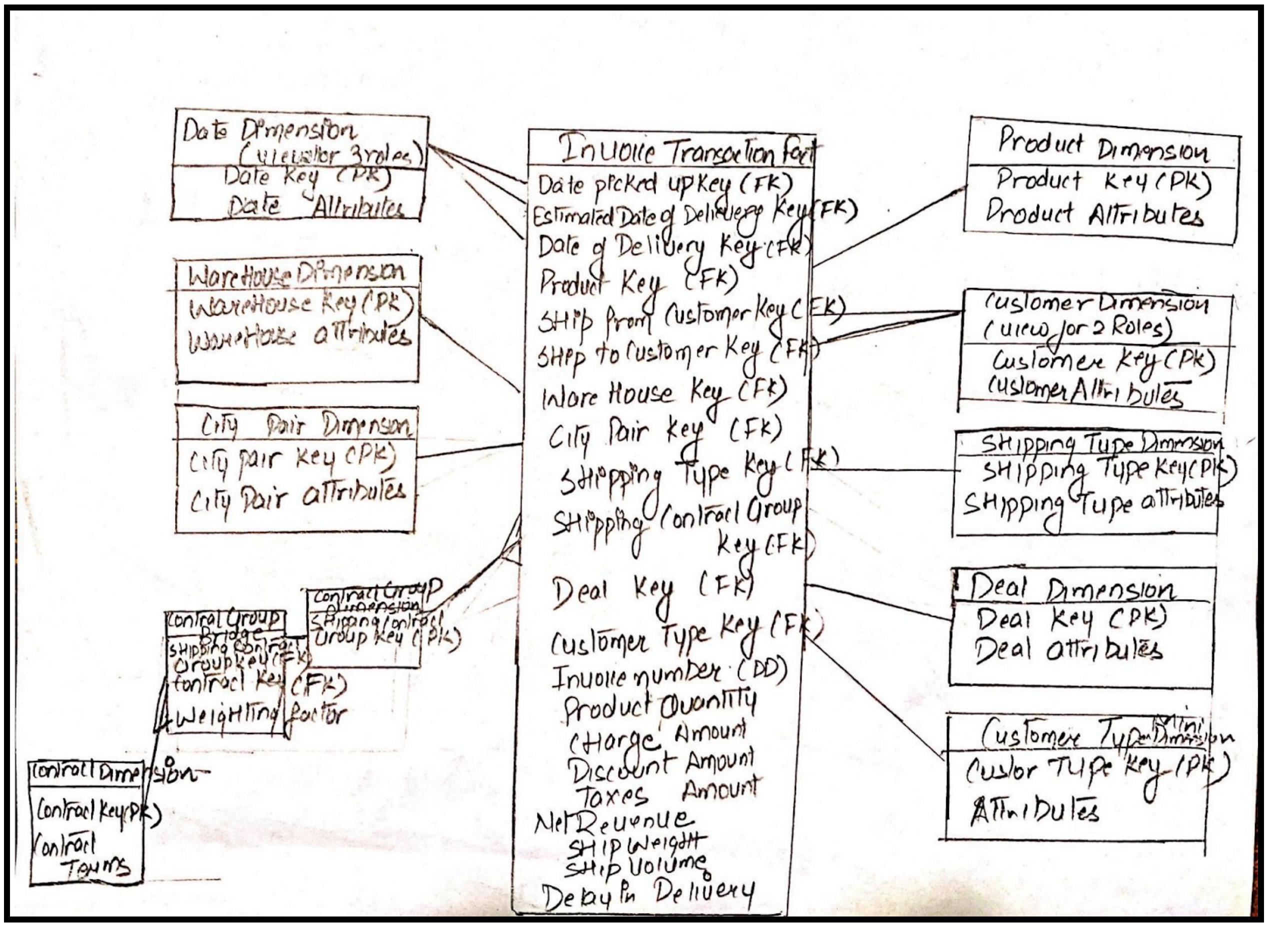
**Name: Ritu(2010644)**

**Part A**

# Star Schema



**Process**- Invoicing transaction

**Grain**-Individual line item on an invoice i.e product.

**Fact keys**- Date Picked up Key, Estimated Date of Delivery Key, Date of Delivery Key,Product Key,Ship from Customer Key ,Ship to Customer Key ,Warehouse Key,City-Pair key,Shipping type Key,Shipment Contract Group Key,Deal Key,Customer Type Key,Invoice number

**Fact measures**- For each row ,Product Quantity,Charge Amount, Discount Amount, Taxes Amount, Net Revenue,Ship Weight, Ship Volume,Delay in Delivery

-Delay in delivery is calculated by taking a difference between Estimated Date of Delivery and Date of Delivery .It can be used to answer queries like 1) How many shipments were delayed for a particular route(city pair) or customer, region, year,shipment mode etc.
This is an important measure of delivery processing efficiency.

Discounts and Taxes are allocated to each row as follows

- Discount for each row (ship discount) will be calculated as

(Charge for that row /total shipping charges for the invoice ) *total discount for the invoice

- Similarly taxes for each row(ship taxes) will be calculated as

(Charge for that row /total shipping charges for the invoice ) *total discount for the invoice

- Net Revenue for each row will be equal to

Charge-ship discount+ship taxes.

**List of Dimensions**- Date dimension with three views-(Date Picked up, Estimated date of delivery , Date of delivery) ,Product dimension,Customer dimension with Ship from Customer view and Ship to Customer view, Warehouse dimension,City Pair Dimension,Shipping Type Dimension,Contract Dimension,Deal dimension,,Customer Type Mini Dimension

**Date- role-playing dimension**

Each column in the date dimension table is defined by the particular day that the row represents. The day-of-week column contains the name of the day, such as Monday. This column would be used to create reports comparing the business on Mondays with Sunday business. The day number in the calendar month column starts with 1 at the beginning of each month and runs to 28, 29, 30, or 31, depending on the month. This column is useful for comparing the same day each month.In our case we are concerned about three different dates- dates picked up, estimated delivery date and delivery date so we place three foreign keys for these. However, we cannot simply join these three foreign keys from the fact table to the date dimension table. SQL would interpret such a two-way simultaneous join as requiring all the dates to be identical, which isn’t very likely.So we create three separate logical tables i.e. views for each date and date dimension is referred to as role playing dimension for playing multiple roles. These three views have unique column names ,for instance, picked up month will be uniquely labeled to distinguish it from estimated delivery month.
Having both estimated and actual delivery date would help us answer queries like
"How many deliveries were more than a week late in the 2nd quarter of 2019?"

**Date Dimension**
Date Key (PK)
Date

Full Date Description
Day of Week
Day Number in Calendar Month
Day Number in Calendar Year
Day Number in Fiscal Month
Day Number in Fiscal Year
Last Day in Week Indicator
Last Day in Month Indicator
Calendar Week Ending Date
Calendar Week Number in Year
Calendar Month Name
Calendar Month Number in Year
Calendar Year-Month (YYYY-MM)
Calendar Quarter
Calendar Year-Quarter
Calendar Half Year
Calendar Year… and more

**Date Picked Up Dimension**
Date Picked up Key (PK)
Date Picked up
Full Date Picked up Description
Date Picked up Day of Week
… and more
Similarly we can have for an estimated delivery date and delivery date with unique column names.

**Customer as a role-playing dimension-**
It is a conformed master dimension with a large number of attributes for each registered customer. The customer dimension rows can play the role of sender of shipment as well as the receiver of shipment. So we create two separate logical tables i.e. views with unique column names . For instance, ship to customer name should be uniquely labeled to distinguish it from ship from customer name. Foreign keys of these two views -ship from customer key and ship to customer key are placed inside a fact table.

**Master Customer dimension**
Customer Key (PK)
Customer ID (Natural Key)
Customer Name

Customer Address
Customer City
Customer State
Customer Zip
Customer Zip Region
Customer  Organization Name
Customer  Corporate Parent Name
…..
And more.
**Ship to Customer View attributes**
Ship To Customer  Key (PK)
Ship To Customer  ID (Natural Key)
Ship To Customer  Name
Ship To  Customer Address
Ship To  Customer City
Ship To  Customer State
Ship To  Customer Zip
Ship To  Customer Zip Region
Ship to Customer  Organization Name
Ship to Customer  Corporate Parent Name
…..
And more.

**Ship from  Customer View attributes**
Ship from Customer Key (PK)
Ship from  Customer ID (Natural Key)
Ship from Customer  Name
Ship from  Customer Address
Ship from  Customer City
Ship from  Customer State
Ship from  Customer Zip
Ship from  Customer Zip Region
Ship from  Customer Organization Name
Ship from Customer Corporate Parent Name…..
And more.

**Warehouse dimension**- There is a possibility of existence of multiple warehouses in one city. In this case knowing only the ship from city won't tell us about the exact warehouse from which a shipment is made and this is important information for us to capture.Therefore we create a separate dimension for the warehouse. It describes every warehouse of the shipping company.

Each warehouse can be thought of as a location. Because of this, we can roll warehouses up to any geographic attribute, such as ZIP code, county, and state in a country. Warehouses can also roll up to store subregion, regions and territory. These two different hierarchies are both easily represented in the  warehouse dimension because both the geographical  and warehouse regional hierarchies are well defined for a single warehouse row. It will hold Warehouse Key (PK), Warehouse ID (Natural Key),Warehouse Name,Warehouse Address,Warehouse City,Warehouse State,Warehouse Zip,Warehouse Zone,Warehouse subregion,Warehouse region,Warehouse territory,Warehouse Total Square Footage… and more.

**Product dimension**- This table provides information about all the  products that are being shipped. It consists of a simple product hierarchy that describes all the shipments' products, including the name of the product, type, brand,category.Products roll up into brands and then into categories. Creating separate tables for brand and category also known as snowflake can make the presentation complex and in turn affects the browsing experience of business users . Therefore we combine them into the product dimension for the ease of use and query performance.
Product dimension will hold Product Key (PK) and other related attributes such as Product Name, Product Description,Product Type,Product Brand,Product Category... and more.

**Deal Dimension-**
**Assumptions-**

1. Each individual product can have a separate deal applicable on it. Therefore a shipment containing multiple products can have multiple deals applied on it.
2. Only one deal can be applied to one product at a time in a shipment . That is one product cannot have multiple deals applicable on it.

Deal dimension describes the deals that have been offered to the customer. It can hold information such as Deal Key (PK),Deal Description,Deal Terms Description,Deal Terms Type Description... and more. There is a one to many relationship between fact and deal. One deal can belong to multiple products in multiple shipments but only one deal can be applied to each row i.e. to each product inside a shipment. However, not all products have a deal applied on them and this results in null being stored in the deal key column. To avoid this situation,one row with value "no deal" is created inside the deal dimension and its corresponding surrogate key is placed inside the fact table for products with no deals.

**Shipment Mode (like air/sea/truck/train)**
**Shipment Class (codes like 1,2,3 which translate to express, expedited, standard)**
Shipment mode and shipping class will be treated as dimensions and not just an attribute as they are the important part of shipment and includes other textual descriptors for the shipping mode and shipping class that interests business users . For instance, Shipping mode dimension may

include attributes such as weight range,size restrictions,proposed delivery time, type of goods it carries(furniture,textiles etc) ... and more. Similarly,shipping class dimension may also include more attributes .In this situation, however, there are only four rows in shipment mode dimension table to indicate air,sea, truck and train modes. Likewise, the shipping class dimension also has just three rows in it, corresponding to express, expedited, standard. Since the row counts are so small,  instead of creating two separate dimension and placing two foreign keys in fact table,we instead combine the dimensions into a single shipping type dimension also called superdimension and place one foreign key for shipping type in fact table.The Cartesian product of the separate dimensions only results in a twelve rows dimension table (four shipping mode rows, three shipping class rows). We also have the opportunity in this superdimension to describe the relationship between the shipping mode and shipping class , such as the shipping group indicator.  This dimension holds attributes like shipping type key(pk),shipping mode,shipping class,shipping group,shipping class code….. more.

**Combined shipping mode-class dimension**

| Shipping type | Shipping mode | Shipping class | Shipping group | Shipping class code | Avg delivery time | More attributes... |
|---|---|---|---|---|---|---|
| 1 | Air | express | Air-express | 1 | 1 day | |
| 2 | Air | expedited | Air-expedited | 2 | 1-2 days | |
| 3 | Air | standard | Air-standard | 3 | 2-3 days | |
| 4 | sea | express | sea-express | 1 | 4-5 days | |
| 5 | sea | expedited | sea-expedited | 2 | 5-7 days | |
| 6 | sea | standard | sea-standard | 3 | 7-9 days | |
| 7 | Truck | express | Truck-express | 1 | 2-3 days | |
| 8 | Truck | expedited | Truck-expedited | 2 | 3-4days | |
| 9 | Truck | standard | Truck-standard | 3 | 4-5days | |
| 10 | Train | express | Truck-express | 1 | 1-2days | |
| 11 | Train | expedited | Train-expedited | 2 | 2-3days | |
| 12 | Train | standard | Train-standard | 3 | 3-4days | |

**Mini dimension**
Business users are interested in slicing and dicing based on the customer type which includes but not limited to the size of a customer and whether they have a good credit status or a high ship frequency or not. Therefore, we opt to create a separate customer type mini dimension, with one row for each unique combination of customer size, ship frequency, and credit status.These are all low cardinality,correlated and frequently analyzed attributes. There would be one row in this mini dimension for each unique combination of size, ship frequency, and credit status in the data, not one row per customer. Business users frequently analyze these columns to select an interesting subset of the customer base. Mini dimension will be directly connected to the fact table by placing the Customer Type key foreign key inside the fact table.Users don't have to go through a large customer table everytime to connect with a fact table for segment level analysis.

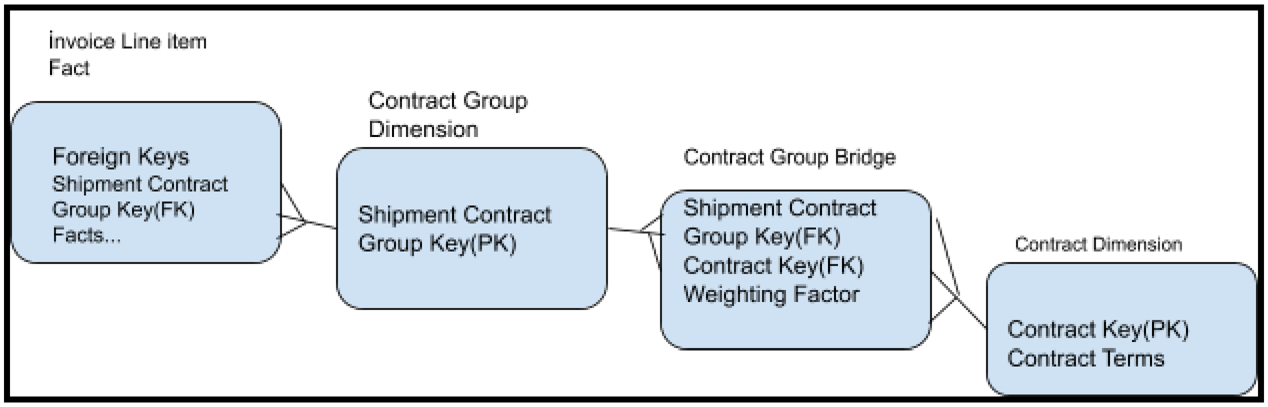
Sample rows

| Customer Type Key | Customer Size | Customer ship frequency | Customer credit status |
|---|---|---|---|
| 1 | small | low | average |
| 2 | small | medium | good |
| 3 | medium | medium | good |
| 4 | medium | medium | excellent |
| 5 | large | high | excellent |
| 6 | large | high | good |

## Invoice number- degenerate dimension
The Invoice number is kept in fact table because it serves as the grouping key for pulling together all the products shipped in a single shipment .It can used to  aggregate invoice line item measurements .It also serves as a primary key along with a product key.It is referred to as a degenerate dimension as we have already extracted interesting header information into other dimensions such as product,date etc.

**Contract** is an important dimension to have that consists of Contract key(PK),contract id along with contract terms.However we are assuming a possibility of a varying number of contracts for each shipment. For instance, one shipment can be a part of 3 contracts while the other one could be a part of 10 contracts and so on. In order to capture this, we replace a contract key from a fact table with a Shipment contract group key. This contract group key is then connected by a many-to-many join to a contract group bridge table, which contains a separate row for each contract in a particular group. Using this table we can identify all the groups for a particular contract.  So, If a shipment is part of four contracts, then that shipment is assigned a contract group with those four contracts. We assign a numerical weighting factor to each contract in the group such that the sum of all the weighting factors in the group is exactly 1.00. The weighting factors can be used to allocate any of the numeric additive facts across the individual contracts. In this way, we can add up all revenue by contract, and the grand total will be the correct grand total revenue. And to get rid of the many-to-many join between fact and contract group bridge table, an extra table whose primary key is contract group is inserted between the fact table and the bridge table. Now both the fact table and the bridge table have conventional many-to-one Join. This table in turn can be used to identify composition for each group.

Contract dimension can be referred to as a **multi valued dimension** as it takes on multiple values (contracts) in the context of the measurement for each row of a fact table .



**Ship From City-Ship To City( City Pair dimension)**
Business users are mostly interested in analyzing how different shipping routes are performing in terms of  revenue generated on route, avg time taken, number of shipments done in the last 3 months or by the distance between cities for a city-pair . To get these attributes that depend on the combination of ship from city  and ship to city, we would create a table  that combines the ship from and ship to city in addition to including the supplemental city-pair attributes .We place a city pair foreign key inside the fact table.Theoretically, the combined dimension could have as many rows as the Cartesian product of the origin and destination city but in real life the number of rows is much smaller than this theoretical limit. This table will give us the city pairs and reflects the way the business thinks about the data.  Attributes- City-Pair key,City-Pair Name,

Ship from city,ship to city… more.

| City-Pair Key | City-Pair Name | Ship from city | Ship to city | Road Distance in miles | Train distance | Air distance | More attributes |
|---|---|---|---|---|---|---|---|
| 1 | SEA-MSP | Seattle | Minneapolis | 1656 | .. | .. | .. |
| 2 | MSP-DFW | Minneapolis | Dallas | .. | .. | .. | .. |
| 3 | DFW-PHL | Dallas | Philadelphia | .. | .. | .. | .. |
| 4 | PHL-JFK | Philadelphia | New york | .. | .. | .. | .. |
| 5 | JFK-LAX | New york | Los angeles | .. | .. | .. | .. |

**Possible SCD-** Slowly changing dimension
1)Customers can have multiple contracts and to accommodate this we create contracts groups and assign those groups to each customer.In this situation, however, the contract group may probably change over time as a type 2 slowly changing dimension (SCD) as customers may opt-out of contracts and sign new contracts. In this case, we would supplement the bridge table with two date stamps to capture begin and end dates for change tracking. This will also allow us to perform time-span queries, such as identifying all customers who were on a given contract at any time between two dates.
2)Another possibility is the customer's location:Customers could change their location which happens occasionally and happens for few customers. So this could act as a slowly changing dimension . We can track this change using type 2 technique that is by adding another row for updated value along with start and end date columns.
**Junk dimension:** There is no junk dimension present in this schema as all the attributes present here are important for invoicing process.

## Possible extensions:

1) **Supertype(Core), Subtype(Custom)** There is a possibility of supertype and subtype in this case.Customers can be of two kinds-Individuals or Organizations. These two types can have some shared attributes as well as some unique attributes.. For instance, individual customers may have attributes defining their first name,last name,marital status,salary etc that's their demographic attributes. However we don't require these attributes for organization.Similarly organizations may have attributes such as organization type,organization department unique to them that are irrelevant to individuals customers . Instead of having all the attributes in one dimension we place common attributes in supertype and unique attributes in subtypes.Supertype customer dimension will have attributes common to both such as address, email ids and flag (indicate whether the customer is an individual or an organization). Etc and the two subtypes -individual and organization will have unique attributes along with id (primary key) that comes from supertype's primary key. These subtypes can be treated as disjoint and mandatory for our process. We create supertype-subtypes to avoid getting large numbers of nulls stored for unique attributes in the customer dimension.This makes joins efficient thereby resulting in faster query execution .

**Supertype- Customer**

| ID | Name | Email | City |
|---|---|---|---|

**Subtype1- Individual Customer**

| Individual ID | Salary | Marital status |
|---|---|---|

**Subtype2- Organization Customer**

| Organization ID | Organization Type | Organization Dept |
|---|---|---|

2) **Row subsets/column subsets**

In cases where business users want to analyze customers based on the shipment mode used such as truck,train,air, sea in our case and if we assume that one set of customers uses only truck and another uses only train i.e. only one of the available modes for shipments. In such scenarios we can create row subsets on the customer dimension to have an independent view of customers using only lets say air shipment and a subset of fact table rows corresponding to those customers. This customer subset will have a customer key as a primary key same as the customer dimension. This provides a direct connection of a subset of customers with a subset of fact without having to go through the huge customer table. This in turn helps make query execution faster.

**3 ) Permissible Outrigger**

Sometimes business users want to perform analysis based on customer tenure or customer

registration month .Customer registration date is required to perform this analysis. To get this date and its attributes we create a copy of the date dimension. This date dimension copy will be semantically distinct from the primary date dimension.All the columns in this copy table will be relabeled as per the registration date.And any constraints on this new date table will have nothing to do with constraints on the primary date dimension table.It will act like a physical copy of the date dimension table called Customer_Register_DATE. This customer registration date view will behave as an outrigger to the customer dimension, also referred to as permissible outrigger.
The customer registration date typically is a join key to this copy.

Another possibility of an outrigger could be the customer's country. If a business is operating across various countries it becomes necessary to track Customer’s country related attributes such as Country capital,Country Continent, Country GDP,Country Per capita … and more as different countries .Country data is available at significantly different grain than the primary customer dimension and is loaded at different times by different sources than the rest of the data.Rather than repeating this large block of data for each customer we will have it as an outrigger. We can save significant space by doing this as the customer dimension is a large dimension.

**If we want to track information about deals in existence on any given date, will the fact table of Invoice line items give us that info? If not, what do we need to add to the model?**

No ,the fact table of the invoice line will not provide accurate results due to the possibility that some of the deals present on a given date were not applied on any shipment made on that day.
So those deals will be missing in a fact table .To capture all the deals including missing deals on a particular date we can create a factless table with two foreign keys- date key and deal key. This table would provide a complete map of the assignments of deals to dates, even if some of the deals were never applied on shipments.

## PART B

a. MiniDimension

Mini-dimension contains the rapidly changing or frequently analyzed attributes of the original huge dimension and are treated as a stand-alone dimension. In a Mini dimension, continuous value attributes such as income, age are converted to banded ranges to maintain a number of rows lower than the original huge dimension. For instance, customer demographics such as age, income are frequently changing attributes present in the customer dimension which can have millions of rows, and in order to track changes, we will create a separate mini dimension with each row for unique combinations of these banded attributes.
We place two foreign keys inside the fact table -one for regular customer dimension and the other one for mini customer demographics dimension. Mini dimensions help us address the

browsing challenges as it provides a smaller point of entry to the fact table. Business users can perform analysis on customers' demographics and access facts without having to go through a huge customer table.  In addition, it helps tackle change tracking challenges.

b. Junk Dimension

A junk dimension is a dimension containing one or more unrelated attributes and is used to avoid having a large number of foreign keys in the fact table. It contains miscellaneous low cardinality flags or indicators unrelated to any dimension and contains the combination of values that actually occur in the data.
For instance, In retail sales transactions , where we have a date,store,product as dimensions we can also have low cardinality attributes such as payment type(cash/credit/check) ,store type(warehouse/marketplace), customer assistance taken flag(yes/no) . These are all unrelated attributes and don't really fit into any dimension but we may need them  as business users may need to query by story type or payment type in future. So we store them in a separate dimension called junk dimension.

c. Degenerate Dimension

It is a dimension that sits inside a fact table and does not have its own dimension table. All related attributes are already placed inside other dimensions. For instance, Invoice number is a degenerate dimension because its attributes such as ship from city, products shipped etc are assigned to other dimensions. And it is left with no attribute. It can be used to group line items for instance to know the list of products that belong to a particular shipment or to calculate total bill amount for each shipment. It also acts as a primary key along with another foreign key such as a product key in invoicing process.
Another example could be a transaction number in a retail sales star schema.

d. MultiValued Dimension

Dimension that takes on multiple values in the context of the measurement for each row of a fact table that is there is possibility of existence of multiple values in a dimension for each row of a fact table. For instance ,in healthcare billing line items,in a single visit, patients can receive multiple diagnoses.  Diagnosis dimension acts as a multivalued dimension in this case.We replace the diagnosis foreign key in the fact table with a diagnosis group key. This diagnosis group key is connected by a many to many join to a diagnosis group bridge table, which contains a separate row for each diagnosis in a particular group. We also place a weight factor column inside the bridge table to allocate facts across each individual diagnosis.

2) A. We can take minidimension from a policyholder dimension or a covered item dimension as both are large dimensions.Policy company can have millions of policyholders and each policyholder can have multiple items covered under insurance.The covered item dimension is usually somewhat larger than the policyholder dimension.

One possible mini dimension could be the risk profile of a policyholder.
Credit Rating(Good,Bad,Average,excellent)
Age Band(18-24,25-35...)
Nature of employment (Travelling,Desk )
Years of Driving experience( 0-3,3-5,...)
Marital status(married/unmarried)
These all are important attributes to determine the risk associated with a policyholder and are low cardinality and frequently queried by business users to analyse revenue generated across different risk profiles. This provides a direct connection to the fact table without having to go through a large policyholder dimension that may contain millions of rows.That's why we create a separate mini dimension .
B.
In this case we would generate custom employee dimension tables one for each type- sales rep,agent,auditor with attributes specific to them because if we attempted to include specific employee type attributes in the employee dimension table, it would have hundreds of special attributes, almost all of which would be empty for any given row.The keys of the custom employee dimensions are the same keys used in the core Employee dimension, which contains all possible employee keys.So a specific employee would be assigned the same surrogate employee key in both the core and custom employee dimensions.No new keys need to be generated; logically, it is like extending existing dimension rows.
We also do not need specific fact tables in this case as there is only one fact being measured which is common to all. We optionally could use these custom-employee dimension tables instead of the core dimension employee table if we wanted to take advantage of the custom attributes specific to that employee  type. These custom tables would be connected to the fact table through the core employee table.